This is a correction to my previous post

While my mockery of the startups that have raised 100s of millions of dollars without shipping a product stays, my thoughts on VLIW and big chips were wrong.

I've been working more on tinygrad, and it does seem that the fundamental operation is not like the computation in software 1.0. From Jim Keller on Lex, there's three fundamental types of compute

CPU: add, multiply, load, store, compare, branch (nothing can be known about anything)

GPU: add, multiply, load, store (when things happen is known, but the addresses aren't)

DSP: add, multiply (everything is known except the data)

Neural networks are DSPs. All the loads and stores can be statically computed, which isn't even possible for GPU workloads, never mind CPU ones.

TBH, I didn't think the problem through well. I assumed that the startups didn't ship because of bad technical tradeoffs; I was primed to believe this from my experience with self driving. I don't think that's the case, and I think they just don't ship because of the general dysfunction in the economy. They don't make chips, they make shares.

For deep learning, VLIW is a good idea. Fancy compilers are a good idea. Because it's not like general purpose compute. Rice's theorem doesn't apply here. In both training and inference, the exact exact same operations are run every time. It's only the data that changes.

I'm kicking myself for getting this wrong. I wrote an accelerator in openpilot based around the idea of the exact same compute every time, and torch 1.10 is adding CUDA caching for this. This same idea applies all the way down the stack.
